2004－IS－90　(5)
2004／11／17

Web アプリケーション統合フレームワークの実装と出張業務支援システムへの適用

福田直樹*　北橋洋三郎**　山口高平***
*静岡大学情報学部
**NTT アドバンステクノロジ
***慶應義塾大学理工学部管理工学科

あらまし：TSIMMIS に代表されるデータ統合のアプローチでは，データの連係に用意されるスキーマの記述力の問題から，データ連係の際にビジネスルール等を介入させることが困難であった．本論文では，データの連係にビジネスルールや人間が介在することが可能な，Ｗｅｂアプリケーション統合フレームワークの実現について述べる．アプリケーション統合の実現に不可欠な，内部共通データ定義を実現するために，アプリケーションの入出力属性とオントロジーを用いた設計支援手法を適用する．本フレームワークを出張業務支援システムの開発に適用し，本フレームワークの有用性を示す．

On Implementing Web Application Integration Framework for Business Trip Support System

Naoki Fukuta*　Yozaburo Kitahashi**　Takahira Yamaguchi***
*Faculty of Informatics, Shizuoka University
**NTT Advance Technology Co.Ltd.
***Department of Administration Engineering, Keio University

Abstract: Data integration approaches proposed TSIMMIS etc., suffers a difficulty of inserting business rule operations and human operations between the data coordinating processes. In this paper, we propose a web application integration framework that enables operations of business rules and humans among the data coordination processes. A design support method for shared data structure definition that plays an important role in web application integration is also proposed. We show the effectiveness of the proposed framework by applying it to business trip support system development.

１．はじめに

これまでに，データ統合に関する研究が行われてきた．その一例としては，TSIMMIS[Chawathe 94]がある．しかし，データ統合では，データを連携させるためのスキーマが用意されるが，そのデータの連係にはビ

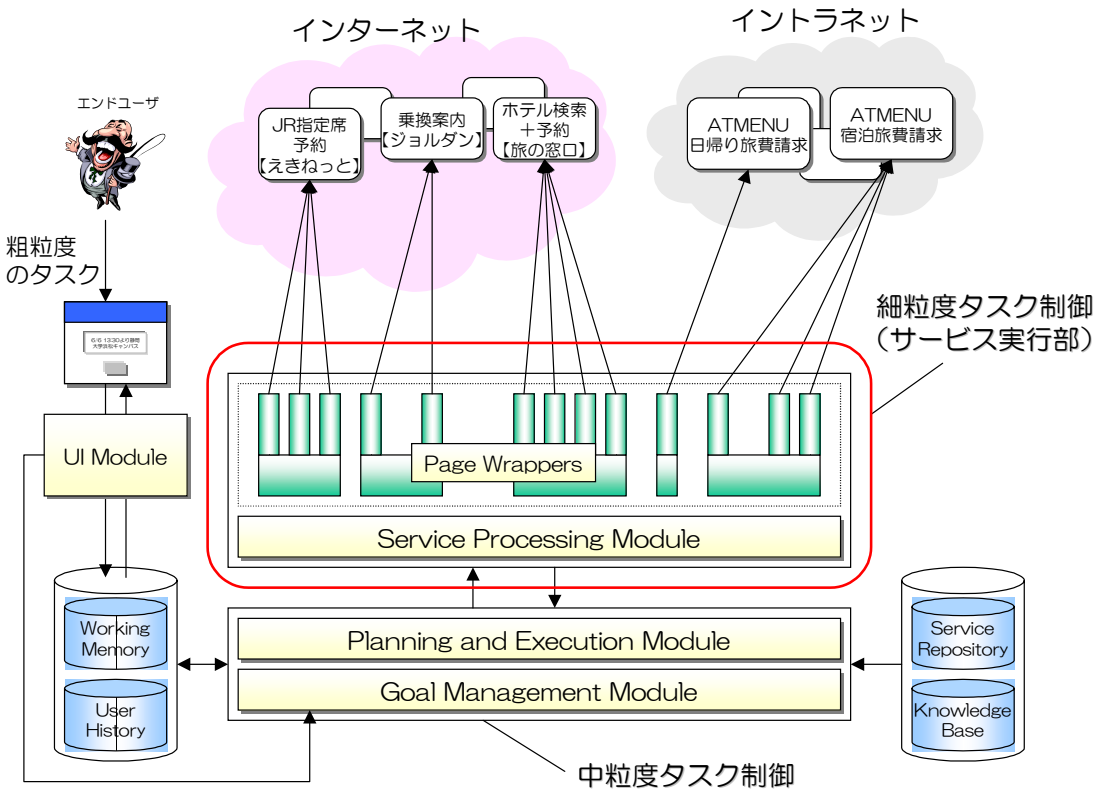

図 1:：Web アプリケーション統合フレームワークの構成

ジネス上のルールや人間の判断を介在させるための仕組みはなく，実世界に副作用を及ぼすようなアプリケーション連携のための基盤とするには不十分であった．本論文では，データ統合のアプローチにおける上述の課題を解決し，アプリケーション統合を実現するためのフレームワークを提案する．アプリケーション統合を実現するためには，データ統合スキーマのかわりにアプリケーション内部で共通的に利用するオブジェクト構造を用意する必要がある．我々は，別途開発した共通オブジェクト構造をクラス図として設計するための手法を用い，共通オブジェクト構造に基づいてアプリケーション間の連携を実現し，その連携にビジネスルールや人間を介在させることのできる枠組みを開発した．さらに，各アプリケーションが提供するサービスを，その共通オブジェクト構造を用いてモデル化することにより，動的なアプリケーション同士の連携を実現可能とした．本フレームワークの有用性を確かめるために，企業内システムとインターネット上の幾つかのサービスを統合した出張業務支援システムを構築した．本フレームワークを用いることで，社内システムとインターネット上のアプリケーションを柔軟に統合できたことを報告する．

２．Ｗｅｂアプリケーション統合フレームワーク

本節では，Ｗｅｂアプリケーション統合フレームワークの設計について述べる．本フレームワークの構成を図１に示す．本フレームワークは，４つのモジュール（UI Module, Service Processing Module, Planning and Execution Module, Goal Management Module）と４つのデータベース（Working Memory, User History, Service Repository, Knowledge Base）から構成される．

Goal Management Module は，ユーザが現

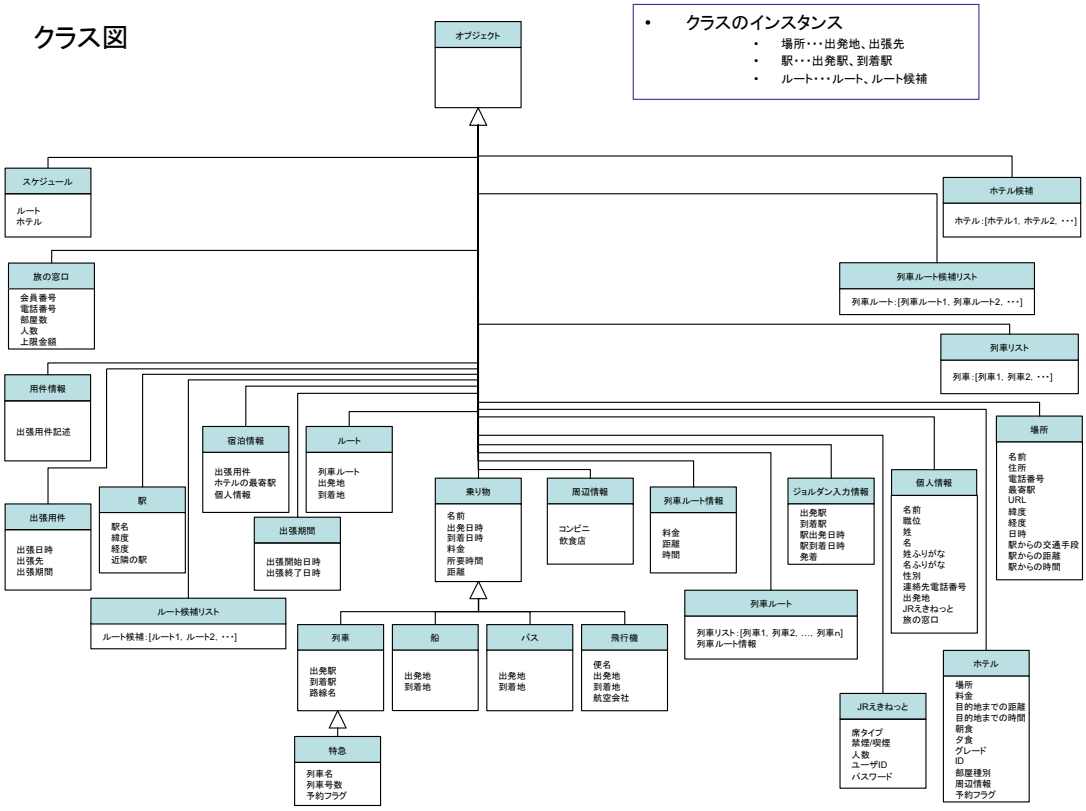

図 2：共通データ定義のクラスの設計

在行っているタスクの最終目的を管理し，ビジネスルールに基づいて，必要に応じて目的の変更をユーザに提示する．たとえば，出張先への電車での当日の移動経路を探索中に，出張先へ時間に間に合うように出発すると早朝に出発する必要が生じる場合がある．例えばこのとき，ビジネスルールで，「往路２００ｋｍ以上の出張の場合には全日の宿泊が可能」というルールがあった場合，そのルールに則って，ユーザに前日に出張先近辺に移動し，ホテルを予約するよう促すことができる．

各モジュール間での通信には，共通オブジェクト定義に基づくオブジェクトの受け渡しが用いられる．これにより，アプリケーション統合におけるデータ連係途上の中間データに対して，ビジネスルールや人間による参照・書き替えおよび統合プロセスの分岐が可能となる．Service Processing Module を独立した構造とし，Ｗｅｂアプリケーションとの通信に Page Wrapper を利用することにより，構成上の汎用性を確保しながら，社内イントラネット上の業務アプリケーションなどの，様々な種類のＷｅｂアプリケーションへの対応が可能となっている．

本フレームワークにより，アプリケーション連携へのビジネスルールおよび人間の介在が可能となるが，アプリケーション連携プロセスをどのように規定するかが課題となる．アプリケーション連携プロセスは，対象とするドメインやビジネスルールに依存する．アプリケーション連携プロセスとフレームワークを分離し，適用ドメインに応じて柔軟に連携プロセスを構築・修正可能とする必要がある．また，状況によってはニーズの個人化等の理由から，あまり頻繁に起きないような連携プロセスが見られる場合があり，そのような連携プロセスの一

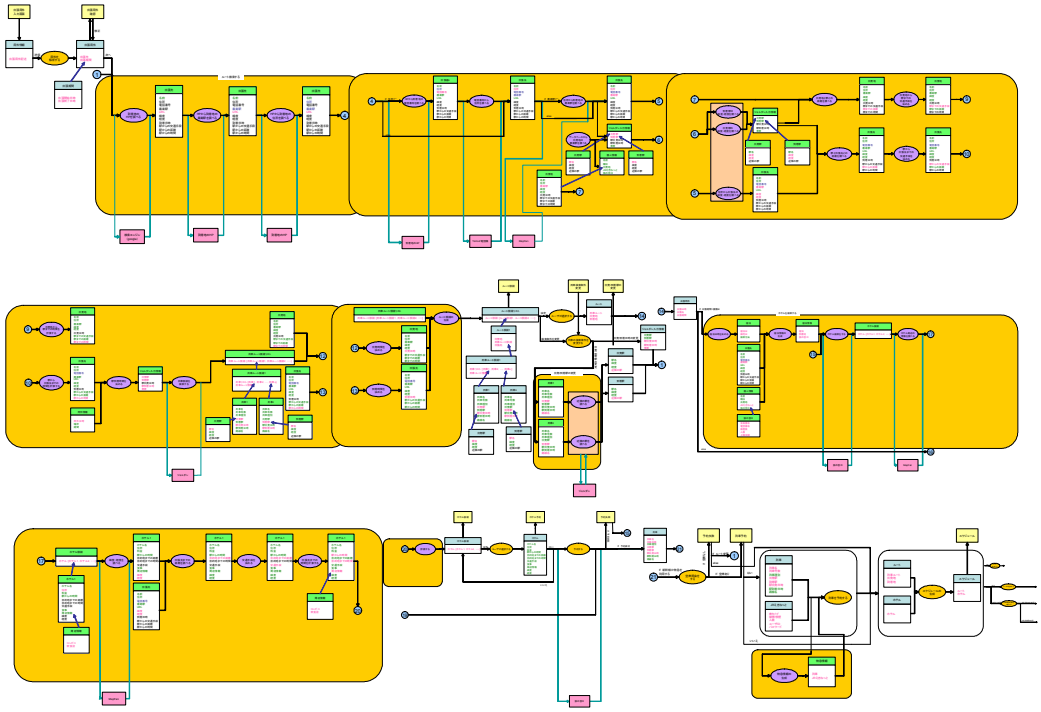

図 3：コアサービス連携プロセスのモデル

般化は難しい．そこで，サービスの連携プロセス自身にも利用者がある程度介入できるための仕組みも構築する必要が出てくる．

3．共通オブジェクト定義の設計手法

本フレームワークを設計するにあたり，各モジュール間でのやりとりに共通オブジェクト定義の存在を仮定している．我々は，文献[Minegishi 04]で，共通オブジェクト定義を，利用するアプリケーション（サービス）の入出力の組と，汎用のオントロジーを用いて構築支援する手法を提案している．本節ではその支援手法の概要を述べる．

本支援手法では，最初に各サービスの持つ入力，出力をそれぞれ 1 つの単位として，初期クラスとする．この初期クラスには，互いのデータの重複や不整合が多数残っている．本誌園手法では，最初に，これらの初期クラスの持つ属性間の包含関係を Lattice 空間上に割り付け，大まかなクラス階層を構成する．ここで，Lattice は全属性集合のべき集合から構成され，下界はすべての属性を含まないもの，上界は全ての属性を含むもので，その間の要素は各属性を含むか否かの組み合わせ集合である．ここで，属性の包含関係に基づいて，初期クラスの親クラスとなるべき候補が提示され，その支援に基づいて人間が初期クラス図から 1 段洗練されたクラス図を設計する．

Lattice 上への割り付けによって洗練されたクラス図は，汎用オントロジーとの関係が比較され，クラスが含む属性の選択方法，上位下位関係の汎用オントロジーとの矛盾等が指摘される．この洗練プロセスを経て，共通オブジェクト定義としてのクラス設計が得られる．

4．出張業務支援システム構築への適用

本フレームワークを，出張業務支援システムの構築に適用した．会社内の出張業務では，会議日程の調整等の出張に依存しない業務の他に，出張先への経路・旅費の調査，旅費の精算のための手続き，出張許可手続き等の煩雑な業務が存在する．従来は，これらの業務を個別の

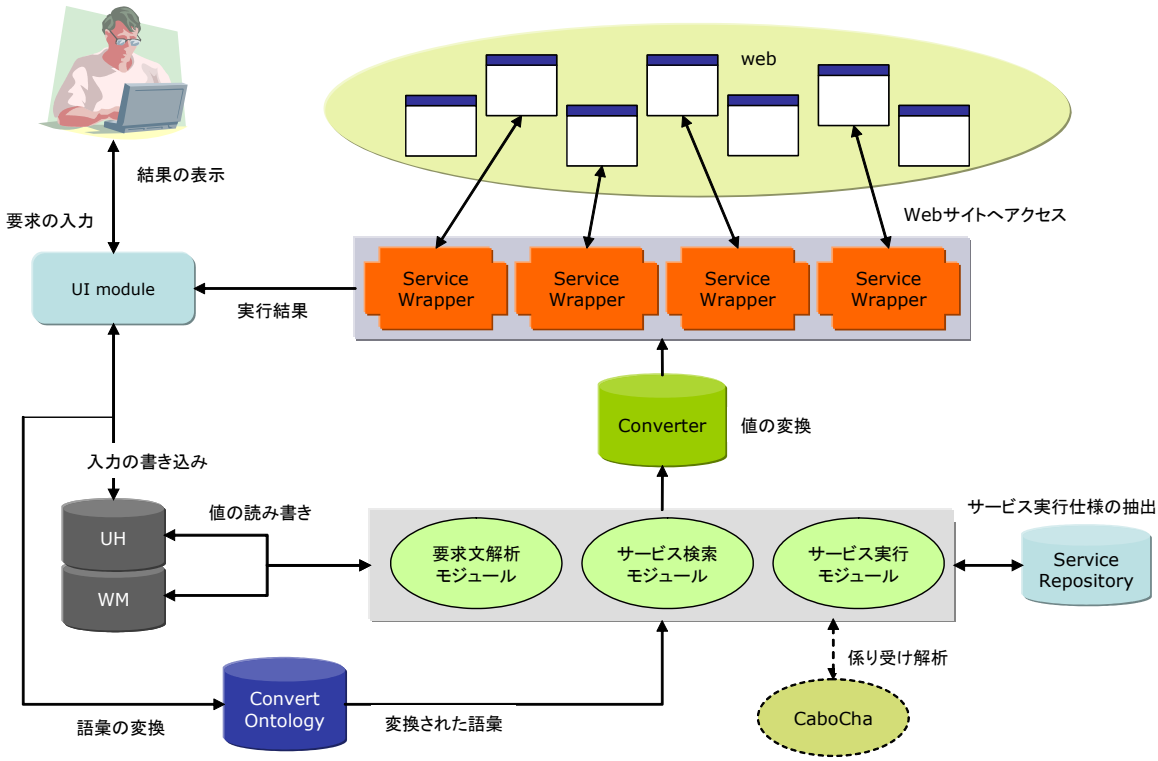

図 4：サービス発見機構

Ｗｅｂアプリケーション等を人間が直接利用することで行ってきていた．しかし，各Ｗｅｂアプリケーションを利用するたびに，出張日程等の共通の情報を何度も入力し直す必要があり，作業の効率化が求められていた．また，インターネット上のＷｅｂアプリケーションには，その社内固有のビジネスルールを考慮しておらず，特定のＷｅｂアプリケーションによって得られた情報がビジネスルールに則っているかを手作業で確認する必要があった．本システムでは，このような出張業務に置ける非効率性を改善し，複数のＷｅｂアプリケーションをビジネスルールに則ってシームレスに利用でき，データの無駄な転記操作を省略可脳とする．

4.1 出張業務支援タスクにおける共通データ定義

システム構築に先立ち，出張業務支援のタスクに利用可能なサービス（Ｗｅｂアプリケーション）を収集した．収集したサービスから，特に出張業務支援タスクに中心的な役割を果たすサービスを選び，それらを元に３節で述べた共通オブジェクト定義の設計手法に基づいて共通オブジェクトのクラス設計を行った．図２が，結果として設計されたクラス図である．本クラス図では特に内部で持つ属性に着目し，メソッドの表記は省略している．また，クラスの継承関係を見やすくするために，クラス間の関係記述を属性記述で代用している．構成されたクラス図は，最上位のオブジェクト型を除き，２６のクラスから構成された．

4.2 出張業務におけるコアサービス連携プロセスのモデル化

出張業務支援における典型的なサービス利用の連携プロセスをモデル化した．図３では，モデル化したプロセスを，ユーザとの対話過程，サービスの呼び出し，およびそこで受け渡され

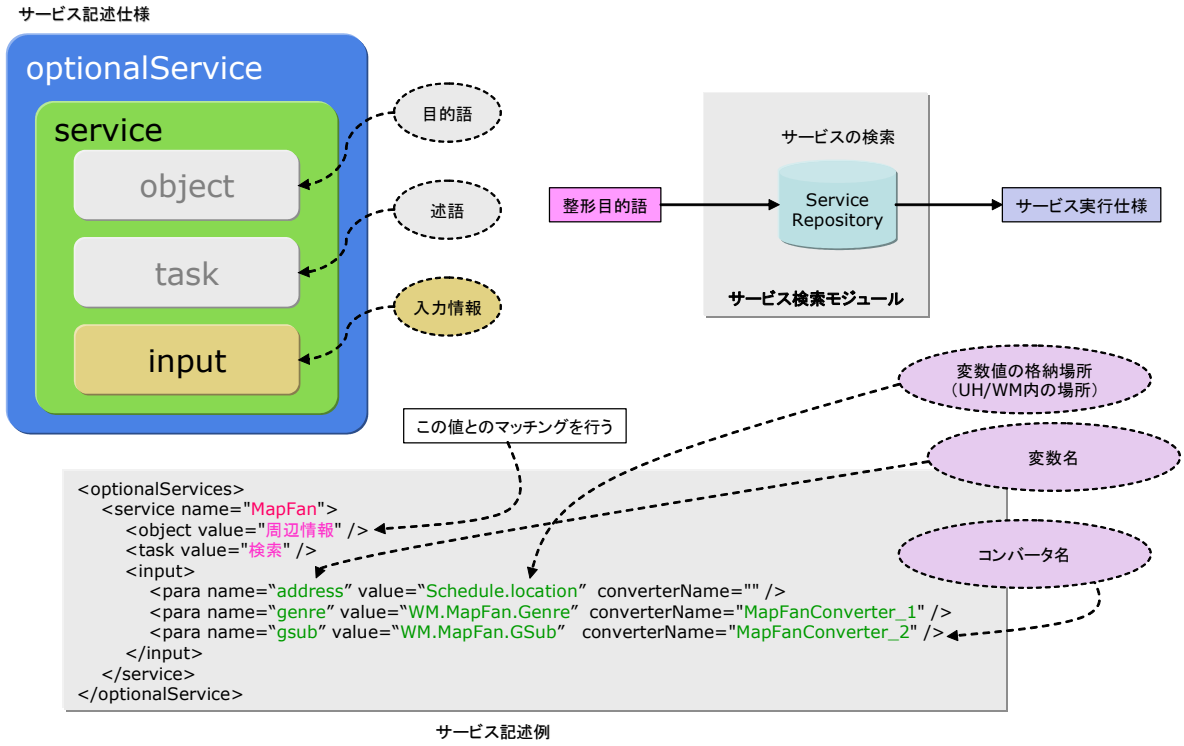

図 5：サービス仕様の記述

るデータの連係プロセスを図示した場合の，プロセスの概観を示している．プロセスは３段にわたった一続きの流れを持っており，各列の真ん中の長四角が１つのプロセスを表現し，その上方向へのリンクがユーザとの対話を，下方向へのリンクがサービスの呼び出しを意味している．図中から，サービスを頻繁に呼び出す部分と，プロセス内部でのビジネスルールに基づく処理を行う部分とが存在することがわかる．

4.3 サービス連携プロセスに基づくアプリケーションの半自動構成

モデル化したコアサービスの連携プロセスに基づき，本フレームワーク上にシステムを半自動で構成できるようにするために，実装コードとモデルとの対応づけを行った．本対応づけでは，１つのユーザとの対話画面，およびプロセス内での処理が，それぞれ１つのサーバサイドプログラムと対応づけられるようにした．通常，サーバサイドプログラムではユーザに提示する画面がプログラムの単位となっており，処理のみを行いユーザに情報を提示しないプログラムは頻繁には用いられない．本フレームワークでは，サービス連携プロセスとプログラムの構成単位を一致させることにより，サービス連携プロセスからのプログラムの構成を用意とした．

4.4 サービス発見機構による動的なサービス連携の実現

4.3 節までで構築したサービス連携プロセスは，典型的な場合に対応できるが，頻度の少ない，個別のニーズに対応するためのサービス連携は実現されない．ユーザからのニーズに応じてサービスを検索し，アドホックなサービスの連携を実現するために，サービス発見機構を用意した（図 4）．サービス発見機構は，サービスリポジトリ内に蓄えられたサービスの入出

力対と，それぞれの入出力属性に対する概念階層を利用し，語彙変換を行うことにより，必要なサービスをリポジトリ内から探し出して実行する．ユーザからの入力は，専用のユーザインタフェースから自然言語文として与えられ，CaboCha[CaboCha]による係り受け解析結果からパターン照合によって検索すべきサービスの要求が取得される．その要求とサービスリポジトリ内のサービスの用途や入出力属性との概念レベルでの照合が行われ，適切と思われるサービスが検索される．次に，WorkingMemory(WM)から，連携可能な一時データが照合され，サービス連携の際に自動で利用される．サービスの呼び出しは Service Processing Module によって行われる．

４．５ サービス仕様の記述

４．３節のサービス連携プロセスの実行，および４．４節での動的なサービス検索と実行において，サービスリポジトリ内にあるサービス仕様記述が参照される．本システムに置けるサービス仕様記述の概要を図 5 に示す．本サービス仕様では，サービスの対象，タスク，入出力属性が記述される．入出力属性には，WorkingMemory に格納される一時データで連係される可能性のある項目が列挙され，値の変換が必要な場合には変換処理機構（コンバータ）が記述される．

４．６ サービス実行部の実現

サービス実行部（Service Processing Module）では，既存のＷｅｂアプリケーションとの相互通信を実現する必要がある．Ｗｅｂアプリケーションとの通信方法には，ラッパーと呼ばれるデータ相互変換機構を利用するが，ラッパーの構成にかかるコストが問題となる．特に，複雑な構造の HTML 文書を出力するＷｅｂアプリケーションや，JavaScript，フレームを利用するＷｅｂアプリケーションへの対応はラッパー構築上の大きな負担となる．

本システムでは，ブラウザの振る舞いを模倣するコンポーネントである HTTP Unit を利用することで，ラッパー構築を容易にしている．本コンポーネントはもともとＷｅｂアプリケーションの動作検証を行うために設計されたものであるが，Ｗｅｂページからの値の取得から，ボタン押下等の操作の模倣までを内部で行うことができる．ラッパー自身は，Ｗｅｂアプリケーションから得られた値の変換処理を記述する必要なため，プログラムとして記述されるが，本コンポーネントを用いることで，開発者は実際のＷｅｂアプリケーション上での操作に近い形でラッパーを記述することができ，ラッパー構築にかかる負担が軽減された．

５ 出張業務支援システムのユーザインタフェース

本システムは，出張先までの列車乗り換え案内，特急列車予約，およびホテル予約サービス等を連携して検索するサービス（コアサービス部）と，コアサービス部で決定した出張スケジュールに関連する他の情報を検索するサービス（オプショナルサービス部）に分かれる．図 6 に，本システムの概観を示す．

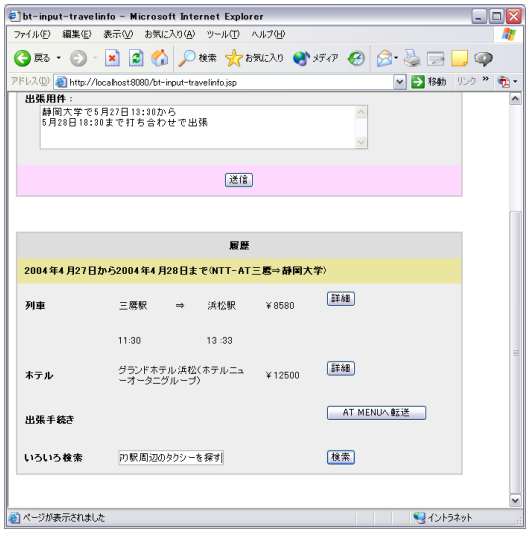

図 6：出張業務支援システムの概観

６．おわりに

インターネット上で提供される従来型のサービスは，そのHTMLベースのインタフェースがユーザと直接インタラクションを行うことを前提としたものであるため，再利用性に乏しく相互接続性はほとんど考慮されていない．したがって利用者はサービス間を“渡り歩く”必要があり，同様の情報を何度も入力させられるなど，エンドユーザに負担を強いる局面が少なくない．

またサービスベンダ／インテグレータの見地からも，サービス間の連携や拡張が容易でないことから，独自のサービス仕様（インタフェース）による顧客の囲い込みを図り，ハイパーリンクを設置するなどの低位の連携に留まる例が多い．このように，インターネット上にはこれまで多種多様なサービスリソースが膨大に蓄積されてきたにも関わらず，それらを充分に活用するためのフレームワークが存在していないのが実情である．

これに対して本システムでは，既存のWebサービスの種々雑多なインタフェースをラッパーによって隠蔽することにより，既存のサービスリソースに高位の相互接続性を追加することを可能にしている．従って，関連する機能を提供していながら，これまでは独自のインタフェースを有していたために連携が不可能であったサービス間であっても，意味レベルのデータを「緩やかに」共有できるようになり，それらがあたかも一つのサービスであるかのようにシームレスに連携を行うことができる．
このようにして，既存の膨大なインターネット資産を利活用することにより，新たなサービスを独自に立ち上げたり，サービス事業者に対してインタフェースの仕様変更を求めることなく，安価で迅速にサービスを開始できることは本研究の事業化時における大きなメリットと言える．またユーザにとっても，これまでユーザ自身が利用してきたサービスが組み込まれていることで，ユーザにとって親和性の高い使用感や安心感を得られるメリットがある．

本論文では，出張業務支援における事例についてのみ述べている．本論文で挙げた事例では，扱うデータが比較的汎用オントロジーに含まれやすいドメインであったため，提案手法がうまく機能したと考えられる．汎用オントロジーでのカバーが難しいドメイン，およびその領域特有のオントロジーが存在するドメインでの提案手法の有効性の検証は，今後の課題である．現在，ゲノム情報学，および社内のレガシーアプリケーション統合に同様の設計手法が適用できないかを検討中である．

謝辞

本研究の一部は，平成１４年～１５年度 新エネルギー・産業技術総合機構大学発事業創出実用化研究開発事業「ユーザのニーズを駆動源としたウェブサービスの動的連携とその流通基盤に関する研究開発」による支援を受けた．

参考文献

[Chawathe 94] S. Chawathe, H. Garcia-Molina and J. Widom. "Flexible Constraint Management for Autonomous Distributed Databases ". *IEEE Data Engineering Bulletin*, Vol. 17, No. 2, pp. 23-27, June 1994.

[CaboCha] http://chasen.org/~taku/software/cabocha/

[Minegishi 04] S. Minegishi, N. Fukuta, T. Iijima, and T. Yamaguchi, ``Acquiring and Refining Class Hierarchy Design of Web Application Integration Software", *Proc. of 5th International Conference on Practical Aspects on Knowledge Management*, Vienna, Austria, 2004 (to appear.)